Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 750–758

Procedia Computer Science

# International Conference on Computational Science, ICCS 2011

# The Prickly Pear Archive

Steven R. Brandt[a,e], Oleg Korobkin[a], Frank Löffler[a], Jian Tao[a], Erik Schnetter[a,c], Ian Hinder[d], Dennis Castleberry[a], Michael Thomas[a,b]

[a] *Center for Computation & Technology, Louisiana State University*
[b] *Department of Computer Science, Louisiana State University*
[c] *Department of Physics & Astronomy, Louisiana State University*
[d] *Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, Postdam, Germany*
[e] *sbrandt@cct.lsu.edu*

**Abstract**

We propose the creation of an on-line journal that integrates with a component framework. By means of the framework, simulations referenced in the paper can be run (or re-run) on journal supercomputers at will, allowing verification of the results or deeper analysis of the paper's problem space. The on-line journal would provide standardized job launch and control mechanisms, data formats, and address code portability issues. Through the use of the framework and its components authors can make use of built-in productivity and readability enhancing features (such as automatic parallelism), leverage code published in previous works, and gain stature as their published modules are used by others. We feel that Cactus is an ideal candidate for the on-line journal, therefore we propose the name "The Prickly Pear Archive" for the on-line journal, taking the name from a species of cactus that can be used to make paper.

*Keywords:* Cactus, framework, executable paper, online journal, portal, component based software engineering

## 1. Introduction

Though there are a number of on-line scientific publications on the web, we draw our inspiration from one of the earliest and most forward-thinking: "Living reviews" [1]. This journal was designed to be regularly updated by its authors and to provide a more interactive experience. Using web content as a format for a paper offers two distinct advantages: navigability and embedded content such as interactive graphs, animations, and portals.

The Prickly Pear Archive (PPA) framework is built upon the Cactus computational framework [2, 3]. In addition to providing the computational capabilities that enable large scale scientific calculations, Cactus provides a platform that help to integrate all the components (called *thorns*, see section 3) required for the PPA system.

Because a given PPA "paper" will consist not simply of text, but of code which can be either re-executed or re-analyzed, it will be possible to use published papers as a platform for deeper exploration. Readers will be able to re-run important calculations with new parameter values. We envision a richer notion of run parameters than simple constants: equations. Modifying an equation parameter would trigger an automatic code generation and a compilation followed by a numerical execution. Automatically-generated code simplifies correctness checking, making it easier for the author to avoid errors and for reviewers to understand the software. This is especially labor-saving for fields such as relativity in which a simple equation can expand to thousands of lines of code. While there have been a number of efforts to generate code directly from mathematical equations, Kranc [4, 5] has the particular distinction of being compatible with the Cactus framework that we are proposing. With Kranc, someone with little or no programming skills could make sophisticated changes and obtain new results.


Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.079

In this paper, we will address various issues in building such an executable paper system. We begin in section 2 with a short introduction of the Cactus computational framework followed by a brief list of all components of the PPA in section 3. We will address executability, validation, copyright and licensing, systems, size, provenance, and finally other issues such as security against viruses and programmability concerns for our executable paper system. Before we conclude, we will present a mock paper that the PPA system will eventually be able to handle in section 4.

## 2. Cactus Computational Framework

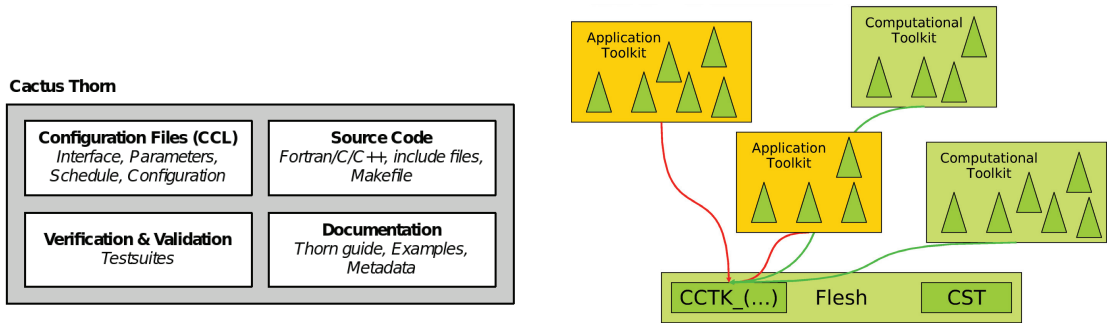


Figure 1: The left diagram shows the internal structure of a typical Cactus module (called a *thorn*). A high-level view of a typical Cactus application is shown on the right diagram, where the Cactus Specification Tool (CST) is used to provide bindings for the Cactus core (called the *flesh and all Cactus thorns. The Cactus Computational Toolkit (CCTK) provides via the Cactus flesh API a range of computational capabilities, such as parallel I/O, data distribution, and checkpointing.*

The Cactus framework [6] was developed to enhance programming productivity and enable large-scale science collaborations. The design of Cactus enables scientists and engineers to develop independent components in Cactus without worrying about portability issues on computing systems. The common infrastructure provided by Cactus also enables the development of scientific codes across different disciplines. This approach emphasizes code re-usability and naturally leads to soundly constructed interfaces and well-tested and well-supported software. As the name *Cactus* implies, the Cactus framework consists of a central piece called *flesh*, which provides an infrastructure and interfaces for modular components called *thorns*. Built upon flesh, thorns provide code for parallelization, mesh refinement, I/O, check-pointing, web servers, and so forth. The Cactus Computational Toolkit (CCTK) is a collection of thorns which provide basic computational capabilities. The application thorns make use of the CCTK by calling the Cactus flesh API (see Figure 1). In Cactus, the simulation domain is discretized using high-order finite differences on block-structured grids. The Carpet AMR library [7, 8] of Cactus enables a basic recursive block-structured AMR algorithm by Berger-Oliger [9]. The time integration schemes are explicit Runge-Kutta methods provided by the Method of Lines time integrator. The Cactus framework hides the detailed implementation of Carpet and other utility thorns from application developers.

## 3. Components of the Prickly Pear Archive Framework

In this section we list the components that form the core of the Prickly Pear Archive, describe them briefly, and identify Executable Paper objectives to which they contribute. This section provides a complementary picture to the following sections in which we list the objectives and explain how they are achieved through the synergy of the components.

The list of objectives we cover maps directly to what is called for by the paper, except that we replace the objective *Other* with *Programmability* and *Viruses*. The former is related to *Validation*, and describes the ease with which authors can make contributions to the archive. The latter describes protection against injection of malware into the archive.

Table 1: Overview of Components of the Prickly Pear Archive.

| Component | Task | Use within the Prickly Pear Archive |
|---|---|---|
| Cactus | *Compatibility, Programmability* | Because Cactus is portable, it addresses the short-term goal of dealing with existing architectures. Through its driver layer abstraction, it makes it possible to adapt codes to future architectures and addresses long-term compatibility. |
| Carpet | *Size, Systems, Programmability* | By providing an easy to use AMR system (addresses *Programmability*), authors are encouraged to make efficient use of memory, disk (addressing *Size*), and CPU (addressing *Systems*) |
| Test CCL | *Validation, Compatibility* | The testing facility built into Cactus makes it possible to check the behavior of a code in a straightforward manner. This makes it simple to identify damage to modules and whether or not their behavior has been changed by new architectures. |
| Formaline | *Provenance* | Formaline will track any changes made to local modules in a simulation and maintain low-level details that might not have made it back to the source tree(s). |
| PetaShare | *Size* | PetaShare provides tools to deal with large data at multiple locations, permitting tasks to be moved transparently to the data. |
| HDF5 | *Size, Compatibility* | The HDF5 file system provides efficient, compressed, and portable storage of grid data in natural hierarchical format. |
| Kranc | *Validation, Programmability* | Kranc is a tool (written in *Mathematica*) which takes equations in preferred form as input and produces code suitable for the Cactus framework. |
| Alpaca Thorns | *Executability, Validation* | The Alpaca thorns enable runtime analysis or steering of a simulation by embedding a web server or Python runtime into the Cactus executable. This makes it possible to understand what is happening in a run as it progresses. This will allow a reviewer to verify that the code is behaving as expected or, alternatively, provide a mechanism for the reader to explore the behavior of a live simulation in greater detail. |
| Dedicated Cloud | *Executability, Validation, Compatibility, Viruses* | *Validation* and *Execution* are the primary purposes of the Prickly Pear cloud. *Compatibility* is also addressed because the details of the operating system install can be preserved and made available at later times. Finally, the use of virtual machines to manage simulations of varying sizes provides a level of insulation against malware (*Viruses*). |
| NRMMA Analysis Framework | *Executability, Compatibility, Validation, Systems* | NRMMA (Numerical Relativity in Mathematica) allows us to directly read compressed HDF5 files that contain visualizable data from grids which are partitioned into multiple levels of refinement. This tool will allow alternative explorations and representations of binary data than those chosen by the authors, and thus addresses *Executability* and *Validation*. It operates on portable binary data (compressed HDF5), addressing *Compatibility*. In many cases the results of a paper will be the result of a "heroic" run–a run too computationally demanding to be re-created by readers or reviewers. In this situation, the flexibility to analyze binary results in new ways may be the only way to address *Systems*. |
| Interactive Plotting software | *Executability* | Figures in traditional journals are fixed images. The PPA could allow interactivity within plots (e.g. zooming in/out, direct manipulations), because it could have access to all information needed to re-create the figure, either from the published data or from executing a similar simulation with changed parameters. |

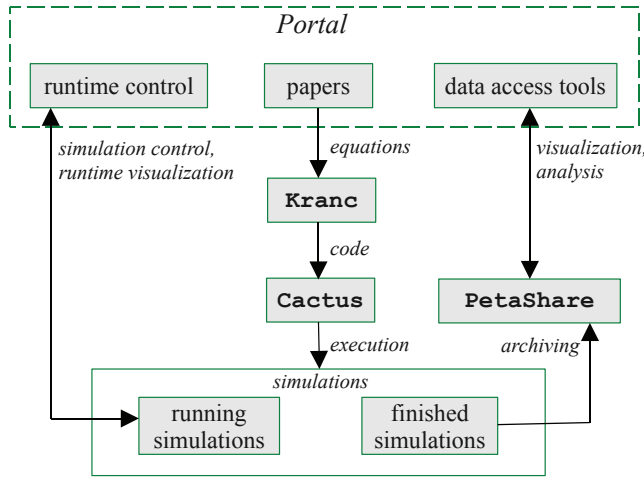


Figure 2: Interaction between different components of the PPA archive system

### 3.1. Executability

Executability of the paper means that most of the equations, tables and graphs in the paper are made interactive in such a way that reviewers and readers can easily check, manipulate and explore the result space. Executability also includes the ability to reproduce numerical experiments and manipulate the results of those experiments. With the availability of source code used to produce published results, it is possible to change key aspects of the simulation by changing the equations, performing a new simulation, and visualizing the result interactively within the on-line paper itself. This enables the reviewer or reader not only to verify the published results, but also to explore the effect of variations of the published parameter values or governing equations.

At least initially, we are planning to use the familiar LaTeX as our base document format. We will, however, provide extensions for extracting equations from a Mathematica source file and converting them to LaTeX, for including content elements such as applets, web *Mathematica* notebooks, flash-based animations, etc. These elements will allow users to rescale plots, change color mappings, provide access to raw data, and so on.

Although dynamic adjustment of plot parameters and the like is easily possible, for long simulations such as those we envisage, dynamic re-computation would not be appropriate. To address this aspect of paper executability, we propose the following design. Each paper on the archive will be linked to the list of simulations on which the paper is based. The list of simulations will be made available to reviewers and readers as they open the paper. Each simulation can be viewed, as well as cloned, modified, and restarted. The thorn Formaline automatically archives the source code of the simulation and stores it in the simulation output directory. Note that the source code does not need to be visible to reviewers or readers for the paper to be executable. Moreover, simulations will have individual user access permissions similar to those in UNIX file systems. Archived simulations or simulations which only have an active text element in the paper describing a specific simulation result or parameter will be linked to that described simulation, which will open as the user clicks on the active text element (see Section 4 below for more details on the interface).

Figure 2 shows the main components of the portal and how the contents of the paper can trigger code execution at different levels all the way down to starting actual simulations on the PPA computational cloud. When a user changes an interactive equation in the paper, this triggers automatic code generation by Kranc which modifies the corresponding Cactus thorn in the current simulation. Restarting the simulation triggers recompilation of Cactus and code execution on the computational cloud. When the simulation runs, the user will be able to steer its parameters and visualize its current data using the runtime control tools on the portal. After the simulation is complete, the user can decide either to discard the data or archive it on PetaShare for subsequent analysis.

#### 3.1.1. Code Generation

One of the key executability features that we propose is the automatic generation of Fortran or C from high-level mathematical expressions. The resulting code is optimized and architecture-aware. The authors of executable papers will be able to specify modifiable equations using the special PPA markup language in their paper source. The PPA engine will then extract the equations in *Mathematica* form that is suitable for processing with Kranc [4, 5] to produce the compilable Cactus modules. The same engine will produce PDF or HTML output, thus ensuring that the equations which are used in the paper are the same as those used in the actual calculations and providing validation.

### 3.2. Validation

The Prickly Pear archive simplifies several aspects of the review process. Above, we describe how the use of code generators allows paper submitters to simplify their development efforts. By making the code generator a part of the core Prickly Pear Archive library, we propose according it the same trust that is given to the output of the Fortran or C compiler. This means that reviewers are freed from the need to verify low-level details of equations.

Generated code will contain the original *Mathematica* formulas in comments, enabling reviewers to verify that the equations in LaTeX form are the ones used by the code.

The source code for the simulation is only one of the necessary ingredients for reproducing the results of a paper. Often, sophisticated analysis must be performed on the data output by the simulation. To fully verify the final results, it is necessary for the reviewer to be able to examine and run this code.

The use of a standard analysis framework built into either the framework or the review system will simplify this process. Such a framework will provide interfaces for reading simulation data (which is usually in a format which is efficient to write during a simulation, rather than one which is easy to read during analysis), performing common analysis operations, and producing the rich and varied plots typically used in papers to present scientific results.

The NRMMA framework was designed to fulfill these needs. NRMMA is capable of reading a set of HDF5 files generated with overlapping levels of various refinement and assembling them for display. NRMMA can generate one-, two-, or three-dimensional data files and supply line and isosurface plots. NRMMA is also capable of sophisticated operations on the data such as extrapolation to infinity, subtracting data at two different resolutions, and displaying convergence.

#### 3.2.1. Reproducibility

Reproducing a simulation is potentially difficult, as details of the operating system, compilers, and installation process can come into play. These difficulties can be alleviated to some extent by the use of virtual machines. At the very least, virtual machines can make the installations behave uniformly.

Because many papers are constructed around heroic runs on big machines, in general it will be impractical to reproduce and interactively modify simulations and their results. In fact, verification will be impossible in many cases. However, it should be possible to construct simulations of similar problems–possibly at lower resolution–which can be carried out on a PPA cloud.

In twenty years, such heroic calculations could be carried out with ease on the PPA. Having a complete specification of the runtime environment would go a long way toward ensuring that scientific results are not lost due to the non-availability of current systems in the future.

### 3.3. Copyright/licensing

Data which was used to publish scientific results should always be freely available to the research community. This also includes tools like simulations software. The PPA naturally offers the possibility to publish all such data. However, authors are often interested in the protection of their intellectual property. This means that authors sometimes don't want to publish the source code or even the executable. The PPA can easily implement various control mechanisms to deny access to sensitive parts of published work. For example, it is possible to re-run simulations without access to the source code or executable of the simulation. In addition, special licenses on works within the PPA could have restrictions on the re-use of data obtained by re-running simulations with modified input parameters.

### 3.4. Systems

In some cases it will be possible to reduce the computational work required to perform a simulation by use of techniques such as AMR. Because the PPA offers access to Carpet, and because the Cactus infrastructure makes the use of this technology so straightforward, PPA researchers should make optimal use of the machines available to them.

Despite this tool, many papers will depend on the data from runs that are simply too computationally expensive for readers or reviewers to verify directly, either for purposes of exploring or validating data.

For some cases of these problems it is possible to use the results of well-chosen sets of expensive runs to provide reliable estimates of results that would be obtained using different parameters [10]. In other cases, repeating results within the context of the PPA will not be possible. However, by keeping reliable records of provenance issues related to these large runs, other researchers with access to large machines should be able to replicate results (when copyright and licensing issues allow).

For cases in which estimates are not possible and large resources are not available, results may be explored and validated with tools that enable analysis of archived binary data. Creating new slices and plots by assembling data in new ways is possible with NRMMA, a framework capable of directly reading and displaying production scale data consisting of overlapping HDF5 files at varying resolutions.

NRMMA also enables subtraction of data sets at different resolutions, making convergence tests possible, allowing readers and reviewers to explore the convergence properties of the data sets in more detail.

### 3.5. Size

Storage and sharing of large files are often concerns for computational science. Cactus thorns can generate output files petabytes in size! However, the size of an individual file in HDF5 (Hierarchical Data Format, version 5) is limited only by the largest-sized integer that can be handled by a compiler. To the great advantage of computational scientists, HDF5 allows applications to be written in different programming languages. This support for a wide variety of programming languages is ideal for the PPA, as authors are not limited by the file format in their choice of a programming language.

What about I/O efficiency? Carpet, an AMR (adaptive mesh refinement) system, adapts the grid resolution of data for optimal storage using the Berger-Oliger formula. When the grid resolution needs to be changed, it is multiplied by an integer factor. Applied to PPA data, Carpet finds the finest grid spacing suitable, thus addressing storage efficiency.

As far as data sharing is concerned, PetaShare, a data-sharing infrastructure which makes use of data-aware storage systems, data-aware schedulers, and a cross-domain metadata scheme, will address the issue of the sharing of large files through a network-accessible storage unit. PetaShare is currently used to manage 700 terabytes of data using the 40 GB/s LONI (Louisiana Optical Network Initiative) infrastructure, making a reliable choice for the PPA's data-sharing needs.

### 3.6. Provenance

Each time a Cactus program is compiled, a module called Formaline is used to store the state of the underlying source code. While the history of module development is saved within a revision control system, this may be insufficient as the code used for any given paper may contain special modifications which are not appropriate for check-in to the revision control system.

There are at least two ways of connecting any given executable with the source code used to build it. The first possibility is to store the source tree in compressed form within the executable. This is presently the default. Another possibility is to store the source code separately from the executable and match two by means of a digital signature.

None of this addresses the issue of registration. Though this need not be known, it can be of advantage – for instance, when a large amount of computational time is involved. It may also be to the user's benefit, as it may be used to store past changes. User registration will be voluntary where it is not made mandatory by the paper authors.

Allowing voluntary registration invites the question: how does one safeguard against abuse through contrived user accounts? How may users be connected with accounts? One of the (in some sense decentralized) systems dealing with a similar issue is the arXiv.org preprint archive. The PPA will adopt a similar technique: one or more endorsements will be necessary to be a user with full rights, including the right to experiment with changes to published simulations.

### *3.7. Security against viruses and code contamination*

The first line of defense will come from the authors themselves. Since all code that will be run on PPA machines will come from researchers whose reputation is potentially at stake, we expect submitted code to be well-scrutinized. However, this by itself is not an adequate assurance that PPA code will be safe against malware.

Because the PPA software permits the triggering of the recompilation of source code by the editing of equations, it will have to be designed with rigorous safeguards. Identifying the correct way to limit the input will be a source of ongoing research. This second line of defense will be handled by the virtual machine on which the PPA will reside. This should serve to minimize the effects of any hostile code submitted to the PPA.

Finally, some forms of code contamination will be identified by the test system (see table 3) or the hashing system described under the provenance section 3.6. Though this may not identify malware, it will protect against contamination through the introduction of bugs.

### *3.8. Programmability*

Programmability is the heart of many issues related to an executable paper. Improving programmability makes code easier to understand and therefore verify. Any executable paper system will need be easy for authors to implement new modules.

The PPA leverages the advantages of both *Mathematica* and the Cactus framework by making it possible to deal with high-level mathematics and simplifying high-performance parallelism and AMR. In both instances, the code becomes more transparent.

## 4. Sample Paper

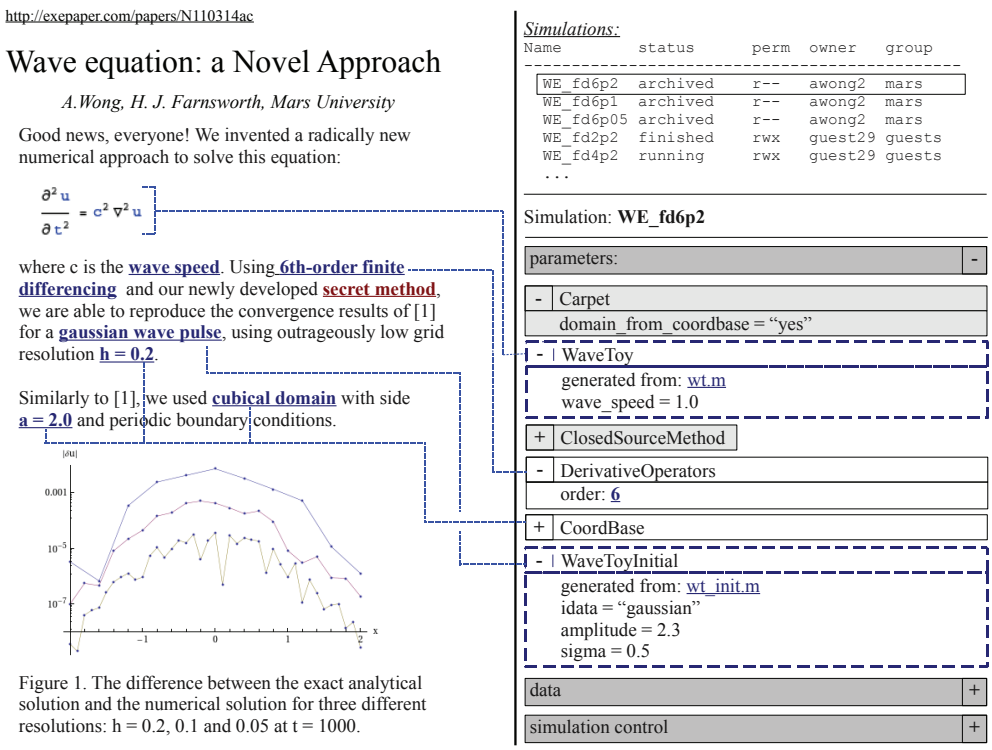


Figure 1. The difference between the exact analytical solution and the numerical solution for three different resolutions: h = 0.2, 0.1 and 0.05 at t = 1000.

Figure 3: This example layout shows what a PPA paper might look like.

In order to minimize the effort required to use the PPA system, we add a set of new property definitions in the LaTeX environment. A potential author can start by writing a paper in LaTeX following the format required by the interested

```
exe-paper
|-- code                                (non-Cactus code if available)
|   |-- exe                             (executables and scripts)
|   `-- par                             (parameter files and other input data if any)
|-- data                                (attached data for graphs)
|-- doc                                 (exe-paper in LaTeX with PPA property boxes)
|-- license                             (licenses if any)
|-- Makefile
`-- sim                                 (Cactus simulations directory)
    `-- WE_fd6p1                        (a sample simulation)
        |-- math                        (Mathematica files for Kranc to generate code)
        |-- plots                       (interactively generated plots)
        |-- anim                        (interactively generated movie clips)
        `-- output-0000                 (simulation restart directory - can be multiple)
            |-- WE_fd6p1.par            (parameter file used for this restart)
            |-- SIMFACTORY              (PPA job management tool)
            |   |-- cactus_wavetoy      (Cactus executable)
            |   `-- ...                 (logged machine-specific metadata)
            |-- WE_fd6p1                (simulation data)
            |   |-- ...                 (data files, essential for publication)
            |   `-- cactus-source       (archived source tree by Formaline)
            |       |-- WaveToy.tar.gz  (WaveToy thorn)
            |       |-- Formaline.tar.gz (Formaline thorn)
            |       `-- ...             (other thorns)
            `-- WE_fd6p1.log            (standard output & error file)
```

Figure 4: This file system tree shows the overall structure of an executable paper in the PPA.

journal. He or she can then make individual parts of the paper (e.g. equations, parameters, initial and boundary conditions) interactive by inserting predefined property boxes in forms of `%property {value}`. These property boxes in the paper will be processed with PPA tools; then the LaTeX code will be transformed into two representations: a traditional printable paper with static figures in postscript format or PDF and an interactive, executable paper in dynamic HTML. In the latter case the code provided by the author can be compiled, executed and monitored. The results can then be visualized and compared to those submitted in the paper. A markup of a sample paper in dynamic HTML is shown in figure 5. The authorized readers will also be able to make slight modifications to the equations presented in the paper and redo the work with a few clicks. Readers will also be able to report breaches of license and copyright infringement via the web interface. The overall file structure of an executable paper in PPA is shown in figure 4.

In the sample paper, `%mathexp {wave3d.evol}` points to a file called `wave3d.evol` that contains a set of evolution equations in *Mathematica* expressions with first order in time. `%par {wave3d.par}` points to a file containing a set of both physical and numerical parameters that could be adjusted to change either the boundary conditions or verify the numerical implementation.

## 5. Conclusions

In this paper, we proposed the Prickly Pear Archive system, an on-line journal integrated with the Cactus computational framework. We addressed various issues towards building such an executable paper system by constructing and integrating multiple components on the Cactus framework. As an example, we present a mock sample paper (see figure 3) that could be handled by the proposed on-line journal.

```
...
%% main equation
The model problem solved is the 3D scalar wave equation in Cartesian coordinates.
$$
\frac{\partial^2 u}{\partial t^2} =
  \frac{\partial^2 u}{\partial x^2} +
  \frac{\partial^2 u}{\partial y^2} +
  \frac{\partial^2 u}{\partial z^2},
$$
which can be rewritten in a set of equations with first order time in time by
defining a temporary variable
$$
  \rho = \frac{\partial u}{\partial t}
$$
%% first order in time that can be used to generate numerical code directly
%mathexp {wave3d}
...
%% running parameters
%par {wave3dpar}
...
```

Figure 5: The excerpt of a mock paper within the PPA shows how LaTeX and *Mathematica* commands are used in a single file. This source will be used to generate the published paper text e.g. in PDF, but also the web pages and paper interface to rerun simulations.

## 6. Acknowledgments

We would like to thank our colleagues Ernazar Abdikamalov, Gabrielle Allen, Matt Anderson, Christian D. Ott, Peter Diener, Tyler Landis, Joel Tohline and Ashley Zebrowski for valuable discussions and ideas. This work is supported by the NSF grants 0721915 (Alpaca), 0904015 (CIGR), 0905046/0941653 (PetaCactus), OCI 0725070 (Track 1 Blue Waters), and the Louisiana NSF/BoR EPSCoR grant CyberTools. When preparing the material for this publication, we used super-computing resources provided by the NSF TeraGrid and Louisiana LONI cyberinfrastructure, especially the computing time allocations TG-MCA02N014 and loni_cactus05.

## References

[1] J. Wheary, B. F. Shutz, Living reviews in relativity: Making an electronic journal live, The Journal of Electronic Publishing, 2007.
[2] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, J. Shalf, The Cactus framework and toolkit: Design and applications, in: Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science, Springer, Berlin, 2003.
[3] Cactus Computational Toolkit home page. [link].
URL http://www.cactuscode.org/
[4] S. Husa, I. Hinder, C. Lechner, Kranc: a Mathematica application to generate numerical codes for tensorial evolution equations, Comput. Phys. Comm. 174 (2006) 983–1004. arXiv:gr-qc/0404023.
[5] C. Ott, E. Schnetter, A. Burrows, E. Livne, E. O'Connor, F. Löffler, Computational models of stellar collapse and core-collapse supernovae, in: Journal of Physics: Conference Series, Vol. 180, Institute of Physics Publishing, 2009, p. 012022.
[6] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, J. Shalf, The Cactus framework and toolkit: Design and applications., in: High Performance Computing for Computational Science - VECPAR 2002, 5th International Conference, Porto, Portugal, June 26-28, 2002, Springer, Berlin, 2003, pp. 197–227.
[7] E. Schnetter, S. H. Hawley, I. Hawke, Evolutions in 3D numerical relativity using fixed mesh refinement, Class. Quantum Grav. 21 (6) (2004) 1465–1488, gr-qc/0310042.
URL http://arxiv.org/pdf/gr-qc/0310042
[8] Adaptive mesh refinement with Carpet, http://www.carpetcode.org/.
[9] M. J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[10] Y. Chen, J. Hesthaven, Y. Maday, J. Rodríguez, Certified Reduced Basis Methods and Output Bounds for the Harmonic Maxwell's Equations, 2009.